Remote Agent Experiment Validation

EXTENDED ABSTRACT

Remote Agent (RA) is a model-based, reusable artificial intelligence (AI) software system that enables goal-based spacecraft commanding and robust fault recovery. RA was flight validated during an experiment on board of DS1 between May 17th and May 21th, 1999.

Technology Overview

RA can operate at different levels of autonomy, allowing ground operators to interact with the spacecraft with immediate commands to the flight software, if needed. However, one of the most unique characteristics of RA, and a main difference with traditional spacecraft commanding, is that ground operators can communicate with RA using *goals* (e.g. "During the next week take pictures of the following asteroids and thrust 90% of the time") rather than with detailed sequences of timed commands. RA determines a plan of action that achieves those goals and carries out that plan by issuing commands to the spacecraft. Actions are represented with tasks that are decomposed on the fly into more detailed tasks and, eventually, into commands to the underlying flight software. When discrepancies are detected between the desired state and the actual state, RA detects, interprets and responds to the anomaly in real time. More serious anomalies can be addressed with longer response times, by generating a new plan of action while the spacecraft is kept idle in a safe configuration. When the new plan is generated, the spacecraft is taken out of the safe configuration and execution resumes normally.

RA differentiates itself from traditional flight software because it is *model-based.* In traditional software programs and expert systems, the programmer decides what the result of a program should be and writes down instructions or rules that attempt to achieve those results. The computer simply executes the instructions or fires the rules with no knowledge of what the intended result was or how it is achieving it. Each component of RA instead operates on *models,* general descriptions of the behavior and structure of the spacecraft it is controlling. Each RA component solves problems by accepting goals and using appropriate reasoning algorithms on its models to assemble a solution that achieves the goals. The reasoning algorithms are general-purpose and remain unchanged across different deployments of RA. For different applications, the parts that change are the models and, possibly, the problem-solving control knowledge needed by some RA modules to tune performance.

Remote Agent Component Technologies

Remote Agent integrates three separate technologies: an on-board planner-scheduler (PS), a robust plan execution system (EXEC), and the Mode Identification and Recovery (MIR) system for model-based fault diagnosis and recovery. These component technologies are described briefly below.

PS—PS generates the plans that RA uses to control the spacecraft. Given the initial spacecraft state and a set of goals, PS generates a set of synchronized high-level tasks that, once executed, will achieve the goals. PS consists of a heuristic chronological-backtracking search engine operating over a constraint-based temporal database. PS begins with an incomplete plan and expands it into a complete plan by posting additional constraints in the database. These constraints originate either from Ground, which imposes them directly on the goals, or from constraint templates (e.g. the camera must be pointed at an asteroid to take a picture of it) stored in a model of the spacecraft. PS queries domain-specific planning experts (specialized software modules such as Deep Space One's navigation system) to access information that is not in its model.

EXEC—EXEC is a reactive, goal-achieving, control system that is responsible for:

- Requesting and executing plans from the planner.
- Requesting/Executing failure recoveries from MIR
- Executing goals and commands from human operators.
- Managing system resources.
- Configuring system devices.
- System-level fault protection.
- Achieving and maintaining safe-modes as necessary.

EXEC is goal-oriented rather than command-oriented. We define a goal as a state of the system being controlled that must be maintained for a specified length of time. As a simple example, consider the goal: keep device A on from time x to time y. If EXEC were to detect that device A is off during that period, it would perform all the commands necessary to turn it back on. EXEC controls multiple processes in order to coordinate the simultaneous execution of multiple goals that are often inter-dependent. In order to execute each goal, EXEC uses a model-based approach to create a complex command procedure designed to robustly achieve the goal.

MIR—The MIR inference engine provides mode identification (diagnosis) and mode reconfiguration (recovery) functionality. To track the state of each component (called a mode) in the spacecraft MIR eavesdrops on commands that are sent to the spacecraft hardware by EXEC. As each command is executed, MIR receives observations from spacecraft's sensors, abstracted by monitors in the spacecraft's control software. MIR combines these commands and observations with declarative models of the spacecraft components to determine the current state of the system and report it to

EXEC. If failures occur, MIR uses the same model to find a repair or workaround that allows the plan to continue execution.

The key idea underlying model-based diagnosis is that a combination of component modes is a possible description of the current overall state of the spacecraft only if the set of models associated with these modes is consistent with the observed sensor values. This method does not require that all aspects of the spacecraft state be directly observable, providing an elegant solution to the problem of limited observability.

Risks

RA is flight software and as such poses the same kind of risks posed by conventional flight software.

The autonomous behavior implemented by RA is not qualitatively different from that displayed by conventional fault protection or attitude control. In all cases, the spacecraft is commanded on the basis of current state information rather than by direct operator commands. The behavior of RA can be predicted, within an envelope, just as the behavior of fault protection or attitude control can be predicted within certain bounds. Confidence in the RA's responses can be obtained through testing, just as confidence in fault protection or attitude control is obtained now.

A risk addressed by the experiment concerns the integration and testing of the technology. RA in a novel integration of three technologies and their application to spacecraft is also new. For this reason there was no prior experience on development and validation methodologies for such a system. Another risk had to do with the integration of the AI technologies of RA, based on general-purpose search algorithms, together with real-time control software on a flight processor.

Validation Objectives

The first validation objective was to demonstrate RA's ability to autonomously operate a spacecraft with communication from ground limited to few high-level goals. This translated into specific objectives for PS, EXEC and MIR. The second validation objective was to show that RA could be commanded with different levels of autonomy. This meant supporting all of the possible operation modes: using EXEC to run a traditional sequence of commands; preparing a plan on the ground and uplinking it to the spacecraft for execution; and providing closed-loop planning and execution on-board the spacecraft. The final validation objective was the first formulation of a development and testing plan for an autonomous flight software system.

Test Program and Results

The Remote Agent Experiment (RAX) consisted of using the RA technology to operate the DS1 spacecraft for several days. We developed a series of operations scenario based on DS1 active cruise mode. In these scenarios RAX commanded a subset of the spacecraft subsystems: Ion Propulsion System (IPS), Miniature Integrated Camera and Spectrometer (MICAS), Autonomous Navigation (NAV), Attitude Control System (ACS) and a series of power switches. The goals in the main scenario were to execute an IPS thrust arc, acquire optical navigation images as requested by the autonomous navigator, and respond to several simulated faults. The faults included minor ones that could be responded to without disrupting the current plan, and more serious faults that required generating a new plan to achieve the remaining goals. We adopted a continuous integration approach in which new features or bug fixes were integrated in new releases only after the integrated system could successfully run the reference scenarios on all available testbeds. We also conducted an extensive formal testing program, separate from the software development process. Testing was distributed on several different platforms of different speeds, level of fidelity and availability to the RA team. Test cases were targeted to the most available testbed that could validate them with the reasonable expectation that the test result would hold on higher fidelity testbeds.

In spite of a couple of bugs that occurred during the flight experiment, RA successfully demonstrated 100% of its flight validation objectives.

Applicability to future NASA missions

The Remote Agent technology is applicable to any future NASA mission that desires or requires autonomous operations. The RA reasoning engines can be used as-is on future missions. New domain models would be required for each mission.